Naan Mudhalvan Phase 3 Assessment

Course Name : Internet Of Things

Project Title : Traffic Management System

Team Name is : Techtronz

Team Members :

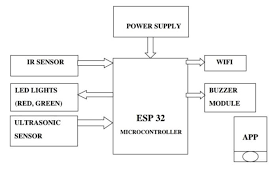
|  |  |  |
| --- | --- | --- |
| Name | Register number | Naan Mudhalvan ID |
| Sanjesh R | 721221106089 | au721221106089 |
| Sivabalvigneshan P | 721221106097 | au721221106097 |
| Sri Krishnan N | 721221106103 | au721221106103 |
| Vishal S | 721221106120 | au721221106120 |

Simulator used:  **Wowki Simulator**

Components used :

|  |  |  |
| --- | --- | --- |
| Component Name | Specifications | Quantity |
| Node MCU | ESP32 | 1 |
| Arduino UNO | - | 1 |
| LED | - | 12 |
| Resisters | 1k | 5 |

Circuit Diagram :



Working :

1 )Hardware Setup:

You will need an ESP32 development board, LEDs (Red, Yellow, and Green for each direction - North-South and East-West), and a 16x2 LCD (1602) with an I2C interface. Connect the LEDs and their corresponding current-limiting resistors to the ESP32 GPIO pins. Wire up the LCD to the ESP32 using the I2C interface (SDA and SCL pins).

2 ) Library Inclusion:

Include the Liquid Crystal library to control the LCD. Ensure you have the ESP32 development board support installed in your Arduino IDE.

3 ) Initialization:

Initialize the GPIO pins for the LEDs as outputs in the setup() function. Initialize the LCD object, specifying the LCD's dimensions (16x2) and its I2C address. Clear the LCD screen and set the cursor position to the top row.

4 ) Main Loop:

The code operates in the loop() function, where it continuously runs. It uses the millis() function to keep track of time and switch traffic directions at specific intervals (60 seconds, in this case).

5 ) Traffic Light Control:

Inside the main loop, the code checks whether the specified duration for a direction has passed. When the duration has passed, it toggles the current traffic direction between North-South and East-West. It turns off all lights and then turns on the appropriate LEDs for the current traffic direction.

6 ) LCD Display:

After changing the traffic direction and controlling the LEDs, the code updates the LCD to show the current direction. It uses the lcd.setCursor() and lcd. print() functions to update the display with the appropriate message based on the direction.

7 ) Repeat:

The code continues to loop, repeating this sequence indefinitely, which simulates the behavior of a traffic signal.

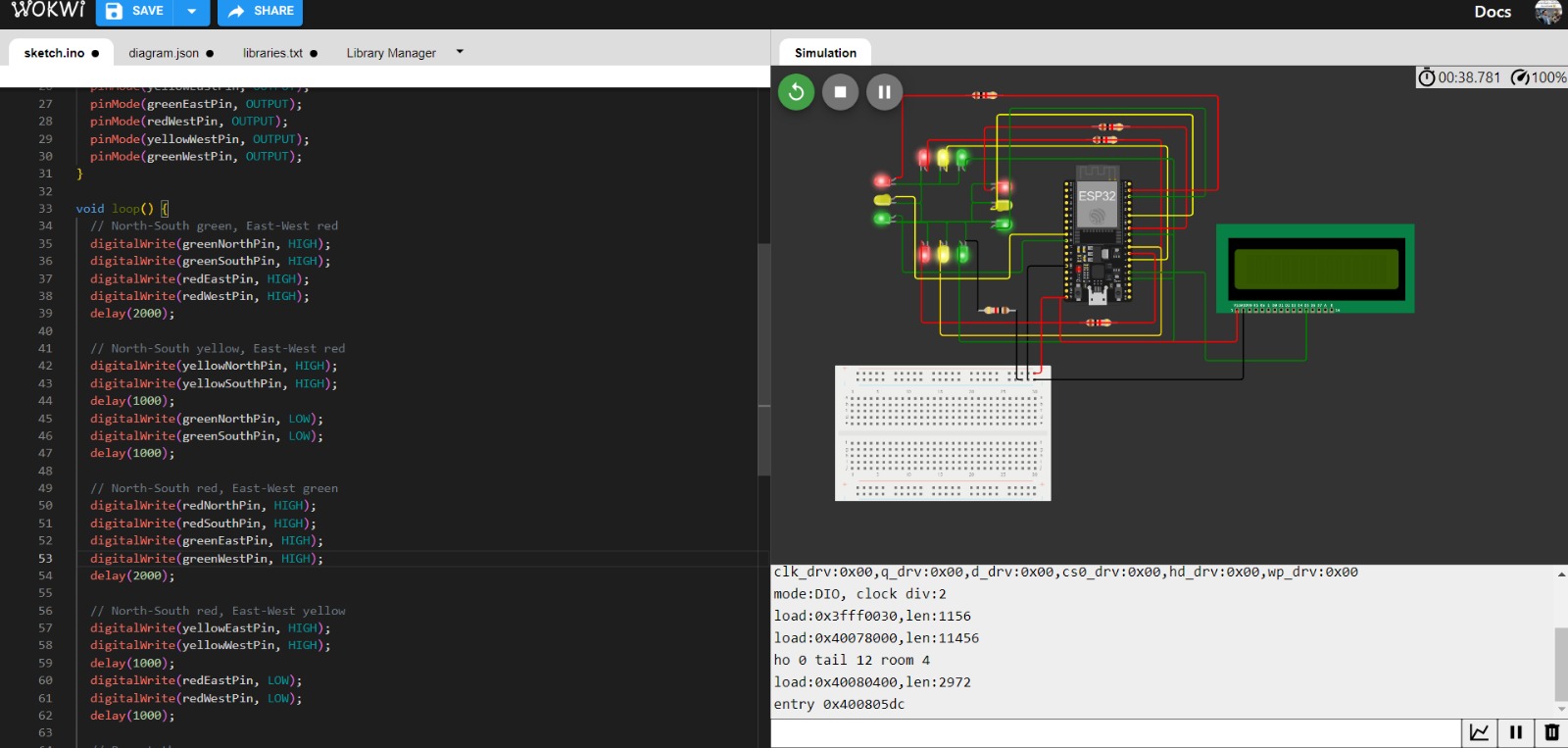
8 ) Upload and Run:

Compile the code and upload it to your ESP32 board using the Arduino IDE or your preferred development environment. Once uploaded, the ESP32 will control the LEDs and update the LCD to simulate the four-way traffic signal with alternating directions.

Conclusion :

In conclusion, the four-way traffic signal simulation project with an ESP32 and an LCD is a practical and educational endeavor that combines hardware and software elements. It provides a foundation for understanding microcontroller-based control systems and is a valuable resource for those interested in exploring the world of embedded systems and real-time control

SIMULATED OUTPUT FOR TRAFFIC MANAGEMENT USING WOWKI SOFTWARE



void loop() {

unsigned long currentMillis = millis();

if (currentMillis - previousMillis >= duration) {

previousMillis = currentMillis;

currentDirection = (currentDirection + 1) % 2; // Toggle between 0 and

turnOffAllLights();

if (currentDirection == 0) {

// North-South green, East-West red

digitalWrite(greenNorthPin, HIGH);

digitalWrite(greenSouthPin, HIGH);

digitalWrite(redEastPin, HIGH);

digitalWrite(redWestPin, HIGH);

lcd.setCursor(0, 1);

lcd.print("NS Green EW Red ");

} else {

// North-South red, East-West green

digitalWrite(redNorthPin, HIGH);

digitalWrite(redSouthPin, HIGH);

digitalWrite(greenEastPin, HIGH);

digitalWrite(greenWestPin, HIGH);

lcd.setCursor(0, 1);

lcd.print("NS Red EW Green ");

}

}

}

void turnOffAllLights() {

digitalWrite(redNorthPin, LOW);

digitalWrite(yellowNorthPin, LOW);

digitalWrite(greenNorthPin, LOW);

digitalWrite(redSouthPin, LOW);

digitalWrite(yellowSouthPin, LOW);

digitalWrite(greenSouthPin, LOW);

digitalWrite(redEastPin, LOW);

digitalWrite(yellowEastPin, LOW);

digitalWrite(greenEastPin, LOW);

digitalWrite(redWestPin, LOW);

digitalWrite(yellowWestPin, LOW);

digitalWrite(greenWestPin, LOW);

}

**Coding:**

#include <Arduino.h>

#include <LiquidCrystal.h>

// Define the LED pin numbers

const int redNorthPin = 2;

const int yellowNorthPin = 4;

const int greenNorthPin = 15;

const int redSouthPin = 16;

const int yellowSouthPin = 17;

const int greenSouthPin = 18;

const int redEastPin = 19;

const int yellowEastPin = 21;

const int greenEastPin = 22;

const int redWestPin = 23;

const int yellowWestPin = 25;

const int greenWestPin = 26;

const unsigned long duration = 60000; // 60 seconds

unsigned long previousMillis = 0;

int currentDirection = 0; // 0 = North-South, 1 = East-West

// LCD configuration

LiquidCrystal lcd(0); // Initialize the LCD object on the I2C interface

void setup() {

// Initialize the LED pins as outputs

pinMode(redNorthPin, OUTPUT);

pinMode(yellowNorthPin, OUTPUT);

pinMode(greenNorthPin, OUTPUT);

pinMode(redSouthPin, OUTPUT);

pinMode(yellowSouthPin, OUTPUT);

pinMode(greenSouthPin, OUTPUT);

pinMode(redEastPin, OUTPUT);

pinMode(yellowEastPin, OUTPUT);

pinMode(greenEastPin, OUTPUT);

pinMode(redWestPin, OUTPUT);

pinMode(yellowWestPin, OUTPUT);

pinMode(greenWestPin, OUTPUT);

// Initialize the LCD

lcd.begin(16, 2); // 16 columns, 2 rows

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Traffic Direction:");

}